

BAKER BOTTS L.L.P.
30 ROCKEFELLER PLAZA
NEW YORK, NEW YORK 10112-0228

TO WHOM IT MAY CONCERN:

Be it known that WE, Armin Amrhein, Johannes Birzer, Martin Kiesel and Regina Schmitt, citizens of Germany, residing at Dresdnerstr. 16, 92245 Kuemmersbruck, Germany, Friedhofweg 2, 92551 Stulln, Germany, Jahnstr. 36, 91099 Poxdorf, Germany, and Herbstaekerweg 5, 91056 Erkangen, Germany, respectively, have invented an improvement in

INTEGRATION METHOD FOR AUTOMATION COMPONENTS
of which the following is a

SPECIFICATION

BACKGROUND OF THE INVENTION

[0001] The invention relates to a method for the integration of a plurality of automation components in a uniform running level model of a respective runtime system of an industrial controller.

[0002] Today, it is customary, both for stored-program control (SPC) and for motion control (MC) systems, to have hierarchical running levels, and to have different software applications for controlling the respective technical processes. The software applications used today may be user-programmed, or they can have built-

in system functions.

[0003] It is known from DE 197 40 550 A1 that process control functionalities of the stored-program controllers and motion functionalities of MC controllers can be integrated in a uniform configurable control system.

[0004] This SPC/MC integration is obtained by using SPC and MC control modules. However, when the integration is carried out in such a way, an optimum and efficient task structure is not achieved for all control tasks. Furthermore, mostly the classic MC functionalities are supported with this type of integration for machine tools, whereas other requirements for the controller, as they are known from the operation of production machines, are not optimally supported by this type of interconnection of SPC and MC control modules.

[0005] In German application DE 19 93 19 33.2 a communication system clock is used between the PC system and the peripheral devices to change between a real-time operating program and a non-real-time operating program. In this case, it is the task of this communication system clock to allow the smoothest possible change between real-time and non-real-time applications in an industrial process. In this configuration, however, the basic clock is only derived from the clock of the communication medium and is only used for changing the operating system mode of a PC system.

[0006] Accordingly there remains a need for the integration of a plurality of

automation components in a uniform running level model of a respective run time systems in an industrial controller.

[0007] An object of the present invention is to create optimum distinctive characteristics of an industrial controller for an industrial controller having different control tasks and different boundary conditions or requirements of the underlying technical process, which provides both SPC and MC functionality or the functionalities of further automation components.

Summary of the Invention

[0008] In order to meet this and other objects which will become apparent with reference to further disclosure set forth below the present invention provides an integration method for automation components in which optimum distinctive characteristics are achieved in principle by a uniform configurable running level model for the control tasks of the industrial controller.

[0009] In one embodiment, the method comprises the following steps:

- a) analyzing the running properties of the automation components;
- b) deriving a structure for a uniform running level model, that satisfies the requirements of the automation components involved;
- c) assigning the system and user tasks into the running level model; and
- d) programming the user level or the user levels.

[0010] This method allows a user to create in a defined, systematic and flexible an

industrial controller with an integrative runtime system which combines process control functionalities of stored-program controllers "SPC", movement functionalities of "MC" controllers or functionalities of further automation components.

[0011] The stratification in system levels and running levels has the effect of minimizing the communication effort between the tasks which represent the functionalities of the different automation components. Furthermore, the programming of the control tasks for different automation components may be done in a uniform programming language with a uniform creation interface. In this way, the user can flexibly create a running level model tailor-made for his respective requirements.

[0012] A further advantage is that user programs can be loaded into the user levels. This allows the user to flexibly adapt the functionality of the automation components to the underlying requirements of the technical process in user programs.

[0013] In another embodiment of the present invention, the system and/or user levels may be prioritized. This increases the flexibility of the user in the creation of an integrative runtime system of an industrial controller.

[0014] In yet another embodiment of the present invention, the basic clock of the running level model is derived from an internal timer or from an internal clock of a

communication medium or from an external device or from a variable which belongs to the technological process. As a result, the basic clock for the running level model can be derived in a flexible and easy manner. Since the basic clock for the running level model also can be derived from a variable which belongs to the technological process, direct feedback from the technological process to the controller easily may be obtained. The user consequently has great flexibility to create the basic clock. The basic clock can, for example, be derived from equidistant bus systems, such as Profibus.

[0015] In still another embodiment of the present invention, user programs are loaded into the user levels. This allows the user to flexibly adapt the functionality of the controller to the underlying requirements of the technical process in his user programs and also to load the user programs into different user levels, in order to achieve effective distinctive characteristics of the controller for respective applications.

[0016] In another embodiment of the present invention, programmed access to the overall functionality of the controller is obtained from the user programs. Consequently, the user can use the controller functionality from the user programs through a uniform interface.

Figure 1 is a block diagram presenting various automation components for operating technical processes;

Figure 2 is an illustrative diagram presenting a hierarchical structure of the main running levels of a classic stored-program controller;

Figure 3 is an illustrative diagram presenting a hierarchical structure of the main running levels of a motion controller;

Figure 4 is a block diagram of an industrial controller;

Figure 5 is an illustrative diagram presenting a hierarchical structure of the integrative running level model of the runtime system of the industrial controller;

Figure 6 is an illustrative diagram presenting the loading of user programs into the user levels; and

Figure 7 is a block diagram illustrating how the basic clock is obtained for the industrial controller.

Detailed Description of the Preferred Embodiments

[0017] Referring to Figure 1, a block diagram of various automation components, 10 (SPC, MC, CNC, RC) for the operation of a technical process (P1, P2; figure 4 and figure 7, respectively) is illustrated. The automation components 110 are connected to one another via a bus connection BV120. It is indicated by 3 dots both at the left end and right end of the bus connection BV120 that there may be other automation components on the bus connection BV120. In addition, the connection to an associated controller unit (industrial controller) or to control units is established by the bus connection BV120. The bus connection BV120 is chosen

only by way of example; other communication media (for example ring connections or wireless connections) are also conceivable. The automation components (SPC, MC, CNC, RC) 110 have different characteristics and different running properties. The analysis of these properties by the user may be carried out by means of heuristic, empirical or other methods of analysis, including a combination of these methods. Response times and cycle times are examples of parameters which are taken into account in this analysis.

[0018] Referring to Figure 2, an illustrative diagram presenting a hierarchical structure of the main running levels of a classic stored-program controller (SPC), arranged according to their priority, is shown. The increase in priority is symbolized in Figure 2 by an arrow 210. In the lowest-priority level 220, as indicated by a dashed line, two different tasks are performed, a free cycle 230, i.e. "user level free cycle" and a background system level 240, i.e. "system level background". The background system level 240 is generally reserved for communication tasks. In a user level, referred to as "user level time-controlled" 250, the calling clock of the tasks or of the programs at this level can be parameterized. Monitoring takes place to ascertain whether the processing of a user program of this clocked level has been completed in time before the start event occurs once again. If the clock time elapses without the user program of the assigned level being processed to completion, a corresponding task of a next-but-one, in priority terms, "user level for asynchronous faults" 260 is started. In this "user level for asynchronous faults"

260, the user can program the handling of fault states.

[0019] The "user level time-controlled" 250 running level is followed by a "user level events" 270 running level . The response to external or internal events takes place within the "user level events" 270. A typical example of such an event is the switching of a binary or digital input, whereby an event is triggered. A "system level high priority" 280 running level includes the tasks of the operating system which ensure the operating mode of the programmable controller (SPC).

[0020] Referring to Figure 3, an illustrative diagram presenting a hierarchical structure of the main running levels of a motion controller (MC) is shown. As with the SPC, the individual levels of the MC are arranged hierarchically according to their priority, as symbolized by an arrow 310. A "system level background" 320 and a "user level sequential" 350 have an equal priority, that is the lowest priority. This common priority level is symbolized as in Figure 2, by a dashed line. The tasks of the "user level sequential" 330 running level are processed together with the tasks of the "system level background" 320 running level in a round-robin procedure. Typical tasks of the "system level background" 320 are, for example, communication tasks. In the "user level sequential" 330, the parts of the program programmed by the user run for the actual control task. If, in one of these parts of the program, the controller encounters a movement or positioning command, a "suspend" is set, i.e., the user program is interrupted at this point. In this case, a command is synchronously used. The processing of this movement or positioning

command takes place in a highest-priority "system level clocked" 340 running level. The position controllers or interpolators which are running in the "system level clocked" 340 running level execute this movement or positioning command. After execution of the command, control returns to the "user level sequential" 330 level and the user program interrupted by "suspend" is continued by a "resume" at the same point. The "system level clocked" 340 level contains not only the already mentioned position controllers but also the interpolation part of the control.

[0021] A "user level events" 350 level resumes at the lowest-priority level. This level accommodates the tasks which respond to external or internal events, such as alarms.

[0022] In a "user level synchronously clocked" 360 level, synchronously clocked user tasks are performed, such as controller functionalities. These tasks are synchronized in relation to clocked system functions, such as the interpolator, position controller or cyclical bus communication.

[0023] A "user level events" 350 running level accommodates those tasks which respond to external or internal events. Such events may be alarms, for example.

[0024] Referring to Figure 4, a block diagram of an industrial controller is illustrated. The control of a technical process P1 is performed by means of the runtime system RTS of an industrial controller. The connection between the runtime system RTS of the controller and the technical process P1 takes place bidirectionally via the

inputs/outputs EA. The programming of the controller, and consequently fixing the behavior of the runtime system RTS, takes place in the engineering system ES. The engineering system ES contains tools for configuring, project planning and programming for machines or for controlling technical processes. The programs created in the engineering system ES are transferred via the information path I into the runtime system RTS of the controller. With respect to its hardware equipment, an engineering system ES usually includes a computer system with a graphics screen (for example display), input aids (for example keyboard and mouse), at least one processor, main memory and secondary memory, a device for accepting computer-readable media (for example floppy disks, CDs) and also connection units for data exchange with other systems (for example further computer systems, controllers for technical processes) or media (for example the Internet). A controller usually comprises input and output units, and also a processor and program memory. It is also conceivable for the control of a technical process P1 to be performed by means of a plurality of runtime systems RTS of industrial controllers.

[0025] Referring to figure 5, an illustrative diagram presenting a hierarchical structure of the integrative running level model of the runtime system (RTS; figure 4) of the industrial controller is shown. The prioritizing of the levels is indicated by an arrow 510 in the direction of the highest priority. The lowest-priority levels are a "cyclical user level" 520 and a "sequential user level" 530. These two levels run

with the same priority. Therefore, these levels are separated in the representation by a dashed line. The "cyclical user level" 520 includes a "background task," which is cycle-time-monitored. In the "sequential user level" 530, the "motion tasks" are run through. The "Motion tasks" are not cycle-time-monitored and serve essentially for describing motion sequences. The "Motion tasks" are processed virtually in parallel. Generally, all the user levels contain one or more tasks. The tasks receive the user programs. The tasks of the "cyclical user level" 520 and of the "sequential user level" 530 are processed in a common round-robin cycle.

[0026] The next-following level is the "time-controlled user level" 540. The tasks of this level are activated in a time-controlled manner. The time control may be set in milliseconds. The "time-controlled user level" 540 is followed by the "event-controlled user level" 550. At this level, after detection of a user interrupt, "user interrupt tasks" are activated. User interrupt events may be formulated as a logical combination of process events and/or internal states.

[0027] The next higher level is a "user level for system exceptions" 560. In the "user level for system exceptions" 560, monitoring is carried out of system interrupts, the occurrence of which has the effect of generating "exceptions", i.e. instances of handling exceptional cases. In the "user level for system exceptions" 560, there are, for example, the following tasks, which are activated when a corresponding system interrupt occurs:

- a) "time fault task", which is activated when time monitors respond;
- b) "peripheral fault task", which is activated, for example, in the event of process and diagnosis alarms, but also in the event of station failure or station return;
- c) "system fault task", which is activated in the event of general system faults;
- d) "program fault task", which is activated in the event of programming faults (for example division by zero);
- e) "time fault background task", which is activated when the cycle time monitoring of the background task responds; and
- f) "technological fault task", which is activated in the event of technological faults.

[0028] The next group of levels is a "synchronously clocked levels" group 570. This group of levels has the highest priority in the running level model. The individual levels of this group of levels may have further prioritization with respect to one another. The group of levels "synchronously clocked levels" 570 comprises at least one system level and at least one user level. The system levels include the system functions, such as, position controller or interpolator. User programs (AP1 - AP4; figure 6) also can be flexibly loaded by a user into the user levels of this group.

[0029] There are a number of different possibilities for clock generation for the clock control of the "synchronously clocked levels" 570. For example, the basic clock

may come from an internal timer (T1; figure 7) or from an internal clock (T3; figure 7) of a communication medium (for example Profibus) or else the clock may also be derived from a process event of the technological process. Such a process event may be, for example, the clock rate (TG; figure 7) of an operation on a production machine or packaging machine. User levels of the group of "synchronously clocked levels" 570 may in this case be clocked on the basis of the basic clock, but they may also run synchronously in relation to one of the system levels of the group of "synchronously clocked levels" 570. The user tasks of this user level synchronous to a system level consequently have a synchronous, i.e. deterministic, relationship with a system level which can be flexibly fixed by the user. This has the advantage that deterministic responses to system tasks (system tasks run in the system levels) which the user has programmed in his user tasks, which run in the user levels of the group of levels "synchronously clocked levels" 570, are guaranteed by the system. That is to say, for example, that the system guarantees that this "synchronous user level" is correspondingly activated for example before the interpolator, or else before any other desired system function.

[0030] The "time-controlled user level" 540, the "event-controlled user level" 550, the "sequential user level" 530, the "cyclical user level" 520 and a "user level for system exceptions" are optional.

[0031] The task of the "cyclical user level" 520 (background task) is cycle-time-monitored. The "motion tasks", on the other hand, are not cycle-time-monitored

and serve essentially for describing sequential sequences. In other words, the present running level model supports a user both in the programming of sequential sequences and in event programming. Consequently, synchronous events and asynchronous events can be covered by the programming. The user programs (AP1 - AP4; figure 6) created by the user also can be loaded into the user levels. The user programs AP1 to AP4 are usually created with the aid of a programming environment of the engineering system (ES; Figure 4).

[0032] Referring to Figure 6, an illustrative diagram presenting the additional loading of user programs into the user levels and distinctive characteristics of user levels of the running level model is shown. There also may be other user levels, or else system levels as indicated by three dots. The prioritizing of the levels is indicated as above by an arrow 610 in the direction of the highest priority. The user levels are assigned the user programs AP1 to AP4, indicated at the right-hand edge of the figure by small squares. The assignment is shown by assignment arrows ZP1 to ZP4. In the user levels there are tasks which receive the additionally loaded user programs AP1 to AP4. These tasks are then run through or processed in accordance with a specific strategy (for example sequentially).

[0033] Referring to Figure 7, a block diagram presenting how the basic clock is obtained for the industrial controller is illustrated. Figure 7 shows by way of example a communication topology into which the controller S is integrated. The controller S is connected by a connection line A2 to the bus B1, to which the

external device EG is attached via a connection line A1. The connection to the technical process P2 takes place via the bus B2. The controller S is connected via the connection line A3 to the bus B2, which in turn establishes the connection to the technical process P2 via the connection line A4.

[0034] The generation of the basic clock of the controller S can take place from different clock sources. For example, from an internal clock source, represented by the internal timer T2 of the controller S or else by an external clock source, such as for example the timer T1, which belongs to the external device EG. The basic clock of a communication medium may also serve, however, as an external clock source. If the bus B2 is realized for example by an equidistant Profibus, the clock for the controller can be obtained from the basic clock of this bus. This is represented in figure 7 by the timer T3 being positioned directly on the connection line A3, and this connection line A3 establishes the connection to the bus B2. The controller is consequently attached to the bus as a slave and can use the bus clock directly.

[0035] There are several variant ways in which the clock for the controller can be obtained from the basic clock of a communication medium (for example a bus). On the one hand, the controller S may be the slave on the bus; the clock information then comes from outside via the bus. On the other hand, the controller S may be the master on the bus. The clock source in this case is provided in the controller S. Two forms with distinctive characteristics exist for this case. The

clock source may lie in a master-bus access circuit or the clock source is in the controller S, in which case the clock is fed into the master-bus access circuit. Furthermore, a clock generator TG which is integrated in the technical process P2 may serve as an external clock source. A clock generator TG in a technical process may be, for example, the operating cycle of a production machine or packaging machine. In the representation according to Figure 7, bus connections are represented by way of example as communication media. However, ring, star or other types of connection may also be chosen as communication media, as well as wireless connections. The basic clock mentioned above can then be derived from these connection systems.

[0036] The foregoing merely illustrates the principles of the invention. Various modifications and alterations to the described embodiments will be apparent to those skilled in the art in view of the teachings herein.

[0037] It will thus be appreciated that those skilled in the art will be able to devise numerous techniques which although not explicitly shown or described herein, embody the principles of the invention and are thus within the spirit and scope of the invention.